

Lab assignment-2.5

Y. Sahasra

2303A51269

Task -1: Refactoring Odd/Even Logic (List Version)

Prompt : Write a Python program to calculate the sum of odd and even numbers in a list Code

and output :

```
File Edit Selection View Go Run Terminal Help <- > Q AI
EXPLORER 1.5 Assignment.py U 2.5 Assignment.py U
1.5 Assignment.py > ...
2.5 Assignment.py > ...
1 #Original code
2 numbers = [1, 2, 3, 4, 5, 6]
3 even_sum = 0
4 odd_sum = 0
5
6 for i in range(len(numbers)):
7     if numbers[i] % 2 == 0:
8         even_sum += numbers[i]
9     else:
10        odd_sum += numbers[i]
11
12 print("Even Sum:", even_sum)
13 print("Odd Sum:", odd_sum)
14
15 # Refactored Code (AI-Improved)
16
17 numbers = [1, 2, 3, 4, 5, 6]
18
19 even_sum = sum(n for n in numbers if n % 2 == 0)
20 odd_sum = sum(n for n in numbers if n % 2 != 0)
21
22 print("Even Sum:", even_sum)
23 print("Odd Sum:", odd_sum)
24
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1 PORTS
TERMINAL
C:\Users\gunda\OneDrive\Documents\Desktop\VAI>c:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/0
--- Test Case 2: Optimized Approach (Slicing) ---
Enter a string: manu
Reversed string: unam
C:\Users\gunda\OneDrive\Documents\Desktop\VAI>
C:\Users\gunda\OneDrive\Documents\Desktop\VAI>
```

Explanation

The refactored code is shorter, more readable, and efficient.

It removes manual loops and uses Python's built-in `sum()` with conditions, making the code easier to maintain.

Task 2: Area Calculation Explanation

Prompt : Explain a Python function that calculates the area of different shapes Code

and output :

```

File Edit Selection View Go Run Terminal Help < - > Q AI
EXPLORER 1.5 Assignment.py U 2.5 Assignment.py U
1.5 Assignment.py > ...
2.5 Assignment.py > ...
17 numbers = [1, 2, 3, 4, 5, 6]
18
19 even_sum = sum(n for n in numbers if n % 2 == 0)
20 odd_sum = sum(n for n in numbers if n % 2 != 0)
21
22 print("Even Sum:", even_sum)
23 print("Odd Sum:", odd_sum)
24
25 #using functions
26
27 def calculate_area(shape, value1, value2=0):
28     if shape == "circle":
29         return 3.14 * value1 * value1
30     elif shape == "rectangle":
31         return value1 * value2
32     elif shape == "triangle":
33         return 0.5 * value1 * value2
34 a=calculate_area("circle",2,2)
35 print(a)
36

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

> > > TERMINAL

```

Odd Sum: 9
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>2.5 Assignment.py
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
12.56

C:\Users\gunda\OneDrive\Documents\Desktop\AI>

```

In 35, Col 9 Spaces: 4 UTF-8 LF () Python 3.14.2 6-8 Go Live

Explanation Gemini clearly explains how the function works for different shapes.

It describes the parameters, logic flow, and formulas used, which helps beginners understand the code easily.

Task 3: Prompt Sensitivity Experiment

Prompt 1: Write a Python program to calculate the sum of even and odd numbers in a list

```

File Edit Selection View Go Run Terminal Help < - > Q AI
EXPLORER 1.5 Assignment.py U 2.5 Assignment.py U
1.5 Assignment.py > ...
2.5 Assignment.py > ...
1 # Program to calculate the sum of even and odd numbers in a list
2
3 def sum_even_odd(numbers):
4     """Calculate the sum of even and odd numbers in a list"""
5     sum_even = 0
6     sum_odd = 0
7
8     for num in numbers:
9         if num % 2 == 0:
10             sum_even += num
11         else:
12             sum_odd += num
13
14     return sum_even, sum_odd
15
16 # Main program
17 if __name__ == "__main__":
18     # Get list of numbers from user
19     user_input = input("Enter numbers separated by spaces: ")
20     numbers = list(map(int, user_input.split()))
21
22     # Calculate sums
23     even_sum, odd_sum = sum_even_odd(numbers)
24
25     # Display results
26     print("Sum of even numbers: (even_sum)")
27     print("Sum of odd numbers: (odd_sum)")


```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

> > > TERMINAL

```

Enter numbers separated by spaces: 1 2 3 4 5 6 7
Sum of even numbers: 12
Sum of odd numbers: 16

C:\Users\gunda\OneDrive\Documents\Desktop\AI>

```

In 14, Col 29 Spaces: 4 UTF-8 LF () Python 3.14.2 6-8 Go Live

Explanation:

For **Prompt 1 (Basic Prompt)**, Cursor AI generated a simple loop-based program using conditional statements. This version is easy to understand and suitable for beginners, but it uses more lines of code and manual variable updates.

Prompt 2: Write a clean and readable Python program to find the sum of even and odd numbers in a list suitable for beginners Code and output:

The screenshot shows a Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled 'AI'. The left sidebar has sections for Explorer, AI, and a file tree showing '1.5 Assignment.py' and '2.5 Assignment.py'. The main editor area displays the following Python code:

```
1.5 Assignment.py U 2.5 Assignment.py U x

2.5 Assignment.py > -
1 numbers = [1, 2, 3, 4, 5, 6]
2
3 even_sum = 0
4 odd_sum = 0
5
6 for number in numbers:
7     if number % 2 == 0:
8         even_sum += number
9     else:
10        odd_sum += number
11
12 print("Sum of Even Numbers:", even_sum)
13 print("Sum of Odd Numbers:", odd_sum)
14
```

The bottom navigation bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the output of running the script:

```
Enter a number: clear
Please enter a valid number.
Enter a number: Traceback (most recent call last):
  file "< c:\Users\gunda\OneDrive\Documents\Desktop\AI\2.5 Assignment.py", line 8, in <module>
    user_input = input("Enter a number: ")
KeyboardInterrupt

c:\Users\gunda\OneDrive\Documents\Desktop\AI\clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gunda\OneDrive\Documents\Desktop\AI>c:\Users\gunda\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/gunda/OneDrive/Documents/Desktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9
```

The status bar at the bottom right shows JavaSE-17 LTS and Python.

Explanation : For **Prompt 2 (Readability-Focused Prompt)**, the AI produced code with clearer variable names and better formatting. Although the logic is similar to the basic version, readability and clarity were improved, making the code easier to review and maintain.

Prompt 3: Write an optimized Python program to calculate the sum of even and odd numbers in a list using built-in functions Code and output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: "1.5 Assignment.py" and "2.5 Assignment.py".
- Code Editor:** The "2.5 Assignment.py" file is open, containing the following Python code:

```
numbers = [1, 2, 3, 4, 5, 6]
even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)
```
- Terminal:** The terminal window shows the following output:

```
KeyboardInterrupt
C:\Users\gunda\OneDrive\Documents\Desktop\AI>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/users/gunda/OneDrive/Documents/Desktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/users/gunda/OneDrive/Documents/Desktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```
- Status Bar:** Shows "Ln 8, Col 1" and "Python" in the status bar.

Explanation: For **Prompt 3 (Optimized Prompt)**, Cursor AI generated a more efficient solution using Python's built-in `sum()` function along with conditions. This version reduced the number of lines and improved code efficiency while maintaining correctness.

Prompt 4 : Write a Python program to calculate the sum of even and odd numbers in a list using functions Code and output :

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows files 1.5 Assignment.py and 2.5 Assignment.py.
- TERMINAL** tab: Displays the command-line output of running the script 2.5 Assignment.py. The output shows:

```
C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/Desktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, and **PORTS** tabs are also visible.
- STATUS BAR**: Shows the current file is 2.5 Assignment.py, line 10, column 1, with 3.14.2 Python version.

Explanation:

For **Prompt 4 (Function-Based Prompt)**, the AI created a modular solution using a user-defined function. This approach improves reusability, debugging ease, and maintainability, making it suitable for larger applications.

Task 4: Tool Comparison Reflection

Reflection

Based on the experiments performed in this lab, Google Gemini, GitHub Copilot, and Cursor AI each have different strengths.

Google Gemini is very useful for understanding code, as it provides clear explanations and works well in Google Colab, especially for beginners.

GitHub Copilot offers real-time code suggestions inside VS Code and is best suited for daily development and writing production-ready code.

Cursor AI is effective for experimenting with different prompts, refactoring code, and analyzing multiple coding approaches.